Refine Search

Search Results -

Term	Documents
(39 AND 13).USPT.	20
(L13 AND L39).USPT.	20

Database:
US Pre-Grant Publication Full-Text Database
US Patents Full-Text Database
US OCR Full-Text Database
EPO Abstracts Database
JPO Abstracts Database
Derwent World Patents Index
IBM Technical Disclosure Bulletins

Search: user response default

Refine Search | Recall Text | Clear | Interrupt

Search History

DATE: Tuesday, August 17, 2004 Printable Copy Create Case

Set Name side by side	Query	Hit Count	Set Name result set
DB=USPT; PLUR=YES; OP=ADJ			
L42	l13 and l39	20	L42
L41	l13 and L40	0	L41
L40	l27 and L39	0	L40
L39	l2 and L38	46	L39
L38	l36 or L37	4600	L38
L37	lowest priorit$	4507	L37
L36	least priorit$	135	L36
L35	l2 and L34	0	L35
L34	least priorit$ same (application$1 or task$1 or program$1)	17	L34
L33	l2 and L32	3	L33
L32	modal state$	22	L32
L31	l16 and l29	2	L31
L30	l6 and L29	3	L30

L29	l2.ab.	265	L29
L28	l2 and l27	18	L28
L27	reclaim$ near3 resource$1	245	L27
L26	718/.ccls.	0	L26
L25	718.ccls.	0	L25
L24	l3 and L23	32	L24
L23	l21 and L22	3512	L23
L22	application near3 (end$ or terminat$)	32090	L22
L21	operating system	65937	L21
L20	l2 same L19	1	L20
L19	default response	155	L19
L18	l2 and L17	3	L18
L17	l13 and L16	100	L17
L16	concurrent processing	1049	L16
L15	conurrent processing	0	L15
L14	cocurrent processing	3	L14
L13	free$ near3 (resource$ or memory or CPU time)	9291	L13
L12	l2 same L11	14	L12
L11	default near3 sav$	390	L11
L10	l2 and L9	3	L10
L9	default save	20	L9
L8	l2 same L6	18	L8
L7	l2 and L6	133	L7
L6	default$ same sav$	2913	L6
L5	l2 and L4	0	L5
L4	l3 same sav$	13	L4
L3	default near3 response$1	680	L3
L2	application$1 near4 (terminat$ or ended)	7314	L2
L1	5784616.pn.	1	L1

END OF SEARCH HISTORY

L42: Entry 8 of 20 File: USPT May 5, 1998

DOCUMENT-IDENTIFIER: US 5748468 A
TITLE: Prioritized co-processor resource manager and method

Brief Summary Text (16):
On an application's request to a device driver for an operation performable on the co-processing platform, the device driver issues a request to load and execute a group of one or more processing tasks to perform the operation on the co-processing platform. In its request to load and execute the group of processing tasks, the device driver associates each requested processing task of the group with a service class for the application-requested operation. Based on their service classes, the co-processor resource manager allocates co-processing resources for processing tasks to be loaded and executed by the platform driver. If co-processing resources for the requested group of processing tasks are previously allocated to currently executing processing tasks, the co-processor resource manager can elect to terminate one or more processing tasks of lower priority to thereby free sufficient resources for the requested group of processing tasks.

Detailed Description Text (21):
In each request to load and execute a node, device drivers 98 assign three attributes--a service class, a real time tag, and an evictable tag--which are associated with that instance of the node for the duration of its execution on co-processor 44. The service class attribute identifies the service class of the operation which the node is being loaded to implement. The real time tag attribute indicates whether the node performs real time processing, and identifies the node as either "real time" or "not real time." The evictable tag attribute identifies the node as being either "evictable" or "not evictable," indicating whether the node can be terminated to free resources for higher priority nodes or whether the node must be allowed to run to completion, respectively. As nodes may be loaded to implement various operations on co-processor 44, the attributes associated with a node, including its service class, may vary between each instance of the node. For example, one instance of a decompression node may be loaded to implement a general purpose audio operation, while a later instance of the same decompression node may be loaded to implement a game operation.

Detailed Description Text (39):
If, however, PCRM 110 determines at step 165 that any resource conflicts exist, the new node cannot be allocated unless sufficient required resources which are currently allocated to nodes having a lower priority can be freed. The PCRM's attempt to free resources for the new node is done in two stages 170 (FIG. 6B) and 190 (FIG. 6C). PCRM 110 first analyzes in steps 172-187 (FIG. 6B) any resource conflicts for singly allocatable resources, before analyzing conflicts for countable resources in steps 192-206 (FIG. 6C). This two stage approach typically is more efficient. Any unavailable singly allocatable resources required by the new node must be freed from the node to which it is currently allocated, or the allocation can not be made. Accordingly, when conflicts for singly allocatable resources cannot be resolved, PCRM 110 can fail allocation of the new node without further analysis.

Detailed Description Text (43):
When the subject node of inner loop 174-183 is lower in priority than the new node,

PCRM 110 then examines the subject node's evictable attribute. Nodes which have been marked as "not evictable" are not subject to prioritized eviction by PCRM 110, and will not be evicted to free resources for the new node. Accordingly, PCRM 110 fails to allocate resources to the new node when the owning node of a resource required by the new node, or any node in the owning node's group, is marked as not evictable. In such case, the resource conflict cannot be resolved because the owning node is not subject to eviction. If, in any iteration of inner loop 174-183, the subject node is not evictable, PCRM 110 fails to load the new node at step 179, and ends method 160 at step 180.

Detailed Description Text (44):
Even when the subject node of inner loop 174-183 is marked as evictable, PCRM 110 also queries for permission to evict the subject node from the DSP platform 94 (FIG. 4) and the client device driver 98 (FIG. 4) which originally requested loading and execution of the subject node. This further query test for evictibility of the subject node allows the platform driver 120 (FIG. 5) or client device driver 98 to override eviction of the subject node. Additionally, through this query test, the platform driver 120 or client device driver 98 is given an opportunity to modify the subject node so as to reduce its resource requirements to avoid eviction. For example, the platform driver 120 or client device driver 98 could reduce the baud rate of a modem node from 9600 bps to 2400 bps, which may free enough resources for the new node without fully evicting the modem node.

Detailed Description Text (47):
When the nodes on the eviction list are evicted to free the singly allocated resources required by the new node, the countable resources of these nodes also will be freed. Analysis of countable resource conflicts therefore can be avoided if evicting the nodes on the eviction list also frees sufficient countable resources to meet the new node's countable resource requirements. Consequently, PCRM 110 first determines whether the addition of the countable resources of the nodes on the eviction list to the available resources also results in sufficient countable resources being available for allocation to the new node. As shown at step 185, PCRM 110 adds the countable resources currently allocated to the nodes on the eviction list to the available resources. At step 186, PCRM 110 compares this sum to the new node's resource requirements. If the comparison shows that sufficient resources would be made available to meet the new node's countable resource requirements by evicting these nodes, PCRM 110 evicts all nodes in the eviction list at step 187, and returns to step 163 (FIG. 6A). Sufficient resources should now be free to meet the new node's resource requirements due to the evictions. Steps 163-166 therefore result in allocation of resources to the new node. If, however, PCRM 110 determines at step 186 that additional countable resources are still needed, PCRM 110 proceeds to analyze countable resource conflicts in second stage 190 (FIG. 6C) of method 160.

Detailed Description Text (48):
In second stage 190, PCRM 110 attempts to resolve the new node's remaining countable resource conflicts by identifying nodes of lower priority than the new node that can be evicted to free needed countable resources. As indicated at step 192, PCRM 110 repeats an outer loop of step 192-203 which traverses the nodes 146 (FIG. 5) currently executing on DSP platform 94 (FIGS. 4 and 5) in ascending order of priority lower than the new node, i.e. from the currently executing node having the least priority to the currently executing node next lower in priority to the new node. In each iteration of loop 192-203, PCRM 110 determines whether evicting the next node in the ascending priority order would free needed countable resources, and whether that node is evictable.

Detailed Description Text (49):
At step 193, PCRM 110 determines whether the next currently executing node in the ascending priority order (hereafter lower priority node) is currently using any countable resources for which the new node has a conflict. PCRM 110 obtains a

resource data structure indicating which resources are allocated to this lower priority node from resource tracker 132 (FIG. 5), and compares it to the new node's resource requirements data structure. If this comparison shows that the lower priority node currently uses any countable resources for which the new node still has a conflict (i.e. the lower priority node is an owning node), PCRM 110 performs an inner loop of steps 194-201 to determine whether such owning node can be evicted to free its resources. Otherwise, if evicting the lower priority node would not free needed countable resources, PCRM 110 continues to steps 202-203 (which result in repeating outer loop 192-203, or if no other currently executing nodes are lower in priority than the new node completing outer loop 192-203 as described below).

Detailed Description Text (50):
Steps 194-201 form an inner loop in which PCRM 110 determines whether the owning node identified at step 193 can be evicted to free its resources for the new node's use. To assess the evictability of the owning node, PCRM 110 repeats inner loop 194-201 for each node (hereafter the subject node in a particular iteration of inner loop 194-201) in a group containing the owning node.

Detailed Description Text (51):
At step 195 of inner loop 194-201, PCRM 110 examines the subject node's (of the current inner loop iteration) evictable attribute. Again, as described above, nodes which have been marked as "not evictable" are not subject to prioritized eviction by PCRM 110, and will not be evicted to free resources for the new node. Further, nodes which are grouped with a non-evictable node also are not evicted. Accordingly, if the subject node is marked as not evictable, PCRM 110 skips all nodes in the group, and exits from inner loop 194-201. Unlike step 175 (FIG. 6B) however, PCRM 110 does not end the analysis of countable resource conflicts when the owning node is not evictable, because portions of the same countable resource can be allocated to many of the currently executing nodes 146 (FIG. 5). The new node's countable resource conflict therefore can still be resolved if another lower priority owning node exists which is evictable. PCRM 110 therefore simply continues to step 202 of outer loop 192-204 on exiting inner loop 194-201 from step 195.

Detailed Description Text (55):
On the other hand, if the comparison at step 202 shows that sufficient resources would be made available to meet the new node's countable resource requirements by evicting the nodes now on the eviction, PCRM 110 evicts all nodes in the eviction list at step 206, and returns to step 163 (FIG. 6A). Sufficient resources should now be free to meet the new node's resource requirements due to the evictions. Steps 163-166 therefore result in allocation of resources to the new node.

Detailed Description Text (58):
Additionally, in alternative embodiments of the invention, the prioritized co-processor resource management method 160 (FIGS. 6A, 6B, and 6C) can be modified to identify an efficient mix of nodes to be freed, not just the lowest priority evictable nodes. For example, when the new node requires countable resources in conflict with a least priority and a next least priority evictable nodes of lower priority than the new node, PCRM 110 in the illustrated method 160 will evict both the least and next least priority nodes if evicting the least priority node does not yield sufficient countable resources to resolve the conflict. If the next least priority node has more of the countable resource in conflict, however, the conflict may be resolvable by evicting only the next least priority node rather than both of the least and next least priority nodes. To identify a more efficient mix of nodes to be freed such as in the example just given, a prioritized co-processor resource management method according to an alternative embodiment of the invention further analyzes combinations of lower priority nodes which own resources in conflict with the new node.

Detailed Description Paragraph Table (2):
TABLE 1 _____ Pseudo-Code Listing of Prioritized

Co-processor Resource Management Method. _____ Get the available platform resources If the new node requires a currently unavailable singly allocatable resource While we can locate a node which uses a required singly allocatable resource For every node in the group containing this node If priority is same or higher than new node's Exit: cannot resolve resource conflict If that node is not evictable Exit: cannot resolve resource conflict Query client and platform for permission to evict If client or platform deny permission Exit: cannot resolve resource conflict If client or platform request a retry Exit: load should be re-attempted Save identity of potentially evictable node For each potentially evictable node identified above Add its countable resources to the available resources If new node requires more of a countable resource than available For each node, from lowest priority to one less than that of the new node If freeing the node would free up a desired resource For each node in the same group If that node is not evictable Skip all nodes in this group Else Query client and platform to evict If client or platform request a retry Exit: load should be re-attempted If client or platform give permission Save identity of potentially evictable node Add its resources to the available resources Else Skip all nodes in this group If available resources now enough Stop looking for more If available resources now enough For each potentially evictable node Evict the node Exit: load should be re-attempted Else Exit: cannot resolve resource conflict

---

CLAIMS:

15. The method of claim 12 comprising:

querying an application which requested an operation implemented by a currently executing node for permission to evict the node; and

if permission to evict the node is denied by the application, excluding such node from being terminated.

18. The method of claim 12 comprising:

decreasing the resource allocation requirements of a currently executing node for an operation having a lower priority service class than the requested operation to free one or more co-processor resources; and

adding the freed co-processor resources to the available co-processor resources.

Previous Doc Next Doc Go to Doc#

L42: Entry 4 of 20 File: USPT Jun 25, 2002

DOCUMENT-IDENTIFIER: US 6412021 B1
TITLE: Method and apparatus for performing user notification

Detailed Description Text (21):
An applet life cycle is primarily governed by the init( ), start( ), stop( ) and destroy( ) methods. The init( ) method is called once after the applet is first launched. The start( ) method is called after the init( ) method and whenever the applet's page is revisited. The stop( ) method is called whenever a user moves away from an applet's page by clicking another button icon in the selection bar. It is suggested that the stop( ) method include steps for saving the state of the applet. The destroy( ) method is called when the desktop manager application exits normally, or when the desktop manager application needs to terminate the applet for other reasons, such as a low memory condition.

Detailed Description Text (23):
In one embodiment, the desktop manager application performs its own memory management. The Java.TM. runtime environment performs garbage collection, but in order for memory to be reclaimed, the desktop manager must release the memory as needed. The desktop manager application continuously monitors the amount of available free memory, which at any given time is in one of three states: "green," "yellow" or "red." "Green" is the normal operating condition. The memory state changes to "yellow" when the system becomes low on memory, and moves to "red" when the system is running out of memory. The thresholds between the states may be set in the desktop manager application configuration files. When the desktop manager application enters the "yellow" or "red" state, the desktop manager application attempts to free up memory by disabling some processes, as well as destroying non-visible images and windows.

Detailed Description Text (32):
In the "red" state, all of the above activities for the "yellow" state have occurred. In addition, the following steps are taken to free memory:

Detailed Description Text (71):
Step 710 represents a branch in execution for a low memory condition. If the desktop manager application becomes aware of a low memory condition, such as described previously for the "red" memory state, in step 711, the desktop manager may call the destroy( ) method of the lowest priority applets that are not currently visible on a display. The applets thus destroyed are effectively unloaded from the Java.TM. runtime environment, in that the memory associated with the applets is released or de-allocated. After handling the low memory condition in step 711, or if no low memory condition existed at step 710, process flow returns to step 704.